On asynchronous dynamic neural field computation

Wahiba Taouali, Thierry Viéville, Nicolas P. Rougier, Frédéric Alexandre

ON ASYNCHRONOUS DYNAMIC NEURAL FIELD COMPUTATION

Wahiba Taouali, Thierry Viéville, Nicolas Rougier, Frédéric Alexandre
INRIA Cortex http://cortex.loria.fr

ABSTRACT

The hallmark of most artificial neural networks is their supposed intrinsic parallelism where each unit is evaluated concurrently to other units in a distributed way. However, if one gives a closer look under the hood, one can soon realize that such a parallelism is an illusion since most implementations use what is referred to as *synchronous* evaluation. The present article propose to consider different evaluation methods (namely *asynchronous* evaluation methods) and to study their properties in some restricted but illustrative cases.

KEY WORDS

CNFT, Dynamic Neural Field, Asynchronous Computation.

1 Introduction

Artificial neural networks rely on distributed computation, but most numerical methods used to resolve the differential equations defining the system evolution require the implicit presence of a central clock. This is indeed true for synchronous computations, in which a central mechanism updates each unit (e.g. neuron model) at the same clock-time. Less obvious is the fact that this is also true for some asynchronous paradigms, in which, for instance, a central mechanism randomly draws without replacement the units to sample at a given regular clock time, in order to simulate asynchronous computation. It means that in computer simulations of physical processes, using explicit numerical methods that calculate the state of a system at a later time from the state of the system at the current time requires complete information about the current state of each of its parts, to deliver from a centralized locus the signal of the next step. For instance, in a clock-based drawing mechanism without replacement, the knowledge of which unit has been sampled or not must be centralized. Worse than that, the paradigm implies that though random sampling events occur at very regular times, it is implicitly assumed that the time is global to the whole system. Computation is distributed, but the computation time and clock remain centralized. At the computational level, this means that if we are using a multi-processor architecture, processors that finish their task early have to wait doing nothing until others finish. The more there are synchronization points, the more performance degrades. At the system dynamics level, the fact that we impose regular updates, even if on a given unit subset, may induce spurious synchronization mechanisms. At the biological modeling level, this means that we assume the existence of a global "universal" clock which is a reasonable approximation for small dynamical systems, but less obvious when considering several cortical maps in interactions with complex connection delays.

In this context, asynchronous mechanism at the mesoscopic level may represent:
- biological delays related to the cortical map topography, with three facets: fixed delay related to known connection length, dynamic delays related to on-going processing or transmission, random delays related to uncertainty or lack of knowledge about the two previous mechanisms;
- local computation effects such as adaptive asynchrony, i.e. the fact that a unit adapts its state with parsimony: the more its value is stable, the less its change has to be output rapidly;
- mesoscopic events such as activity synchronization, rhythms, or sudden activity change.

As soon as such semantics is targeted, there is no place for a central mechanism to decide "a-priori" which unit is going to update its state, whereas each unit has to calculate on its own, both what is its next value and when its next value is going to be updated.

Since it is rather counter-intuitive to rely implicitly on such a centralized scheme, we would like to study to which extent we can remove this central clock assumption and implement a really decentralized (asynchronous) computation. This has been already studied in the case of cellular automaton [14, 13, 15, 4] and parallel computations [5, 6] using particularly Discrete Event Systems Specification (DEVS) to simulate discrete time systems and approximate, as closely as desired, differential equation systems. The DEVS (Discrete Event Systems Specification) formalism [28] provides a way of expressing discrete event models and a basis for an open distributed simulation of dynamic environments in which "events" occur. It also supports hierarchical modular construction, such as microscopic neurons, mescoscopic columns, etc...Using DEVS abstractions to capture the spiking nature features of biological neurons that were not represented in conventional artificial neural networks, started with Pioneer works of [26, 24], exploiting these capabilities to perform intelligent control tasks.

We review in this paper some mathematical models of asynchronous computation, showing some convergence results and illustrating the use of these models in dynamic neural fields computation, including time discretization problems. The purpose is not to give a complete state-of-the-art on asynchronous models, but to show how a gen-

eral construct, called here "fully asynchronous paradigm", provides a constructive answer to asynchronous computation, especially in the particular case of artificial neural networks. Thus, we briefly present in section 2 some computational models dealing with asynchrony aspects in discrete and continuous dynamic systems. Then, we focus in section 3.2 on the bias induced by the discretization of continuous dynamic systems, allowing us to explain the foundation of well-defined asynchronous computation regarding dynamic neural fields, in section 3.3. Finally, we propose, in section 4, to use an event-driven paradigm, as an adequate framework to simulate an intrinsically asynchronous system, before concluding this work.

2 Computational asynchronous models

2.1 Discrete dynamic systems

A discrete dynamic system is a finite set of elements, each taking a finite number of states evolving in a discrete time, by mutual interactions. In [19], which is a book dedicated to the analysis of the temporal dynamics of such systems, Robert introduces what he called "a chaotic discrete iteration mode". The studied system is a cellular automaton with N Boolean cells (a finite number of states), which may correspond, in a neural network, to an active neuron state (spiking) or a silent state. Each unit is influenced by a subset of units that are involved in its update. Starting from an initial state at $t = 0$, at each time step each unit updates its state function involving the states of the corresponding subset of elements, using a specific iteration mode (parallel, serial or chaotic). In a parallel mode, at every time step all the units are updated simultaneously. In a serial mode, at each time step units are evaluated one by one respecting at every iteration the result of the previous one (Gauss-Seidel method). This introduces a kind of partial asynchrony in the system but the central clock is still needed, so it is still macroscopically synchronous. Thus, we are interested in the chaotic mode. It is assumed that there is still a discrete clock but that only serves to number the events. So we move from a time-driven paradigm to an event-driven paradigm. At each event occurrence, only a subset of arbitrarily chosen units is evaluated and it is supposed that there is no transmission delays. The main problem, even if the system is finite and time is discrete, is to ensure the convergence to a stable state. In the numerical analysis community, the reference book in both continuous and discrete context is Bertsekas and Tsitsiklis book, chapters 6 and 7 [6], released in 1989. It presents some algorithms and gives sufficient conditions for convergence for general nonlinear problems and necessary conditions for linear problems. Here, Robert introduces the notion of pseudo-period. A pseudo-period corresponds to a sequence of events in which each unit is updated at least once. The main result presented in [19, 3], is that if a system is pseudo-periodic, it converges at most after N pseudo-periods. So with only N synchronization points the system convergence is guaranteed. But when we introduce transmission delays, such an assumption is no more valid, thus does not ensure the convergence.

2.2 Continuous dynamic systems

Although discrete dynamical systems are a powerful modeling formalism, it is insufficient for modeling systems that have physical components. Physical systems are usually modeled by differential equations in continuous dynamical systems. We are going to focus on Mitra works [17] dealing with asynchronous relaxations for numerical solution of differential equations. In parallel processing like in neural computation, we may have different run times and units are not supposed to wait for each other. Additionally, transmission delays contribute to desynchronize the exchanged information. As described in [17], both phenomena are obvious to take into account in a "fully asynchronous" paradigm. An important discussed point is that to prove the convergence of asynchronous algorithms, two main assumptions are required. First, the delays should be bounded by a finite constant d. Then, a non starvation condition is required. Each unit should be updated at least once in all the sequences of completion that would be of arbitrary but fixed length, with $length < s$. This joins the previous model results. Different versions of these assumptions have been assumed in asynchronous computational algorithms like in [11]. Therefore, with some additional conditions on the differential equations system (specially the leak function), uniform convergence at a geometric rate is proved. The main idea, as shown in section 3.2, is that an unbiased discretized implementation of a continuous dynamical system must take into account for each simulation time, a sampling relaxation scheme (i.e., generating several samplings for a given simulation time) in order to bound the error along the system trajectory.

3 Neural network asynchronous computation

3.1 Dynamic neural field (DNF)

The system is a *network* of *units* with *connections* between units. Each *unit*, at the mesoscopic biological level, corresponds to a cortical column (see, e.g., [12] for a discussion on the concept), the *network* is a cortical map (see e.g. [25] for a discussion on the concept), and may be modeled by a dynamical neural field [27, 2, 23], generating temporal *events* (e.g, synchronization, rhythms, or sudden activity change). Therefore, the evolution of a neural population activity is described by the following differential equation (see [21] for details):

$$\tau \frac{\partial V(\mathbf{x},t)}{\partial t} = -V(\mathbf{x},t) + \int_M w(\mathbf{x}-\mathbf{y}) f(V(\mathbf{y})) d\mathbf{y} + h + s(\mathbf{x},t) \quad (1)$$

where $\mathbf{x}$ denotes a location onto the manifold M; t is time; $V(\mathbf{x}, t)$ denotes the membrane potential of a neural population at point $\mathbf{x}$ and time t; τ is the temporal decay of synapses, f is a sigmoid function computing the mean firing rate, w is a neighborhood function, $s(\mathbf{x})$ is the input received at position $\mathbf{x}$ and h is the mean neuron threshold.

3.2 Continuous versus discrete modeling

When symbolic resolution is not possible, the evolution of such a system can be approximated using numerical integration, e.g. low order methods such as Euler-forward method or higher order methods such as Runge-Kutta method [18].

Let us consider the very simple case of a linear constant approximation of the system, written:

$$d/dtV_j(t) + L_j(t)\, V_j(t) = \sum_{k \neq j} W_{jk}(t)\, \sigma\left(V_{jk}(t)\right) + I_j(t), \tag{2}$$

with initial condition $V_j(0)$, in the particular case where leak L_j, connection strength W_{jk} and current input I_j are constant, while $\sigma\left(u\right) = u$ (see [1] for a discussion of the kind of "sigmoid" profiles usually used) . This writes in vectorial form:

$$d/dt\mathbf{V}(t) = -\mathbf{A}\, \mathbf{V}(t) + \mathbf{I},$$

with

$$\mathbf{A} = \begin{pmatrix} L_1 & -W_{12} & \cdots \\ -W_{21} & L_2 & \cdots \\ \cdots & \cdots & \cdots \end{pmatrix},\ \mathbf{I} = \begin{pmatrix} I_1 \\ I_2 \\ \cdots \end{pmatrix},$$

and its regular sampling forward Euler discretization writes :

$$\mathbf{V}[i+1] = \mathbf{V}[i] - \Delta T\, \mathbf{A}\, \mathbf{V}[i] + \mathbf{I},$$

at $t = i\Delta T$.

Here, we have to assume that the system is *contracting*, i.e., that real part of the eigen-values of $\mathbf{A}$ are strictly positive, otherwise the system does not converge towards a stable solution, and the Euler-forward approximation method is not expected to converge towards a continuous solution (see e.g. [18] for these elementary notions). In words this means that leak is strong enough with respect to the weights in order to induce the system convergence, see [1] for a detailed study in the case of discrete neural fields. More precisely, on an eigen-direction (i.e. in the direction of an eigen-vector of the matrix), the linear equation is decoupled from the others and the leak (either a real or a complex value) corresponds to the opposite of the eigen value, with solution either damped oscillations or an exponential vanishing profile. We do not have to assume that weights are symmetric, but that the matrix $\mathbf{A}$ is diagonalizable, which is always the case up to a negligible singular set, not taken into account here. In the non-linear non-constant case, if the system is hyperbolic, the same condition applies on the Jacobian of the system at any time and state value.

In such a simple case it is obvious to study[1] both the continuous scheme and its discrete approximation starting from the same initial value converge towards the same fixed point (which can be found in all text books), but not though the same trajectory (which is surprisingly not studied in text books up to our best knowledge). More precisely the bias in an eigen-direction of the $\mathbf{A}$ matrix is proportional to $V_j(0) - I_j/\lambda$, where λ is the eigen-value (i.e., the leak) in this direction, and follows a double exponential profile, only function of $\Delta T\, \lambda$, as illustrated in Fig. 1. In words, the highest $\Delta T\, \lambda \in [0, 1[$ (this boundaries corresponding to the convergence interval), the highest the bias magnitude, but the quickest the bias between both methods vanishes. The highest leak thus determines the maximal bias, the smallest leak the maximal duration of bias.

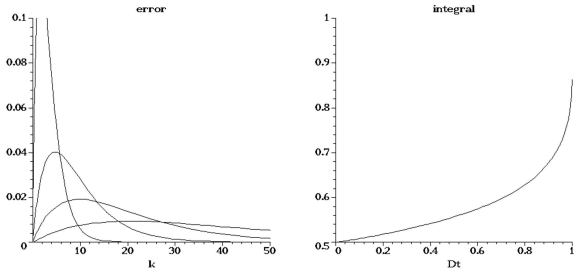

Figure 1. *Left view*: The normalized bias temporal profile between the continuous scheme and its discrete approximation, drawn here for $\Delta T\, \lambda = [0.05, 0.1, 0.2, 0.5]$ from the flattest to the sharpest curve respectively. *Right view*: The integral of the bias along the trajectory as a function of $\Delta T\, \lambda$, making explicit that the cumulative bias is never negligible even for very small leak, while it diverges for large leak. See text for details.

In the non-constant case (i.e., leak, weights or currents vary with time), the previous results generalize considering bounds of the, now variable, leaks. In the non-linear case, the previous results generalize bounding the non-linear function $\sigma()$ by the corresponding maximal slope linear function. See, e.g. [9] for a review of such tools.

[1] In the scalar case, an explicit closed-form is automatically derived from a few lines of, e.g., `maple` symbolic code:

```
eq := D(V)(t) = -A * V(t) + b:  assume(0 < A, A < 1):
# Continuous solution, assuming t0 = 0
s_c := dsolve(eq, V(0) = V0, V(t));
# Euler approximate integration, assuming delta_t = 1
s_e := rsolve(subs(eq, t = k, V(k + 1) = V(k) + Dt * D(V)(t)), V(0) = V0, V);
# Bias analysis
err_k := simplify(factor((subs(s_c, t = Dt * k, V(t))
- subs(s_e, V(k))) / (V0 - i / A)), Dt * A = c);
```

while the result is straightforward to apply to the eigen-value decomposition of the $\mathbf{A}$ matrix. Furthermore, in the scalar case, if the Euler approximation is used with $Dt = (1 - exp(-A\,\Delta T))/A = \Delta T - A/2\Delta T^2 + O(\Delta T^2)$ in numerical scheme, the bias is canceled, which is not generalizable in the vectorial case since it depends on the leak value.

It is a counter-intuitive and very important result to notice that large leaks (i.e. small time-constants) indeed accelerate convergence, but with the drawback to generate large errors during the first iterations. This means that the system dynamics may very easily switch from one attractor to another, at the beginning of the trajectory, even in such a very simple case.

Furthermore, the cumulative bias is never negligible, with a lower bound for small leak, while diverging for large leaks. This shows the very important difference between the fact that the discretization methods *converge at least* toward the expected fixed point and the fact that the simulation trajectory is unbiased. In this simple example, unbiasedness never occurs, except in the singular case where $V_j(0) = I_j/\lambda$.

Where stands the "mistake" ? At the implementation level, it stands on the simple fact that there is a confusion between the *sampling* time (i.e., the time at which the continuous system is discretized) and the *simulation* time (i.e., the simulated dynamical system time). We obviously need several sampling times for a given simulation time in order to make the discrete approximation converge towards the continuous one, which is often not taken into consideration. This is going to be developed in section 3.3

At the modeling level, it stands on the belief that a pertinent model of the reality has to be a "continuous" model, its discretization being a kind of second-class implementation detail. This is definitely wrong when modeling digital computational systems, but this is also questionable for microscopic neural models (see, e.g., [10] for a discussion on biologically plausible generalized integrate and fire neuron models) and mesoscopic neural map models (see, e.g. [21] for a discussion at this modeling scale), the key question being "what do we want to learn" from the model or its simulation. This is going to be discussed in section 4.

3.3 Asynchronous computation

Following [17] and instantiating its paradigm in the case of a dynamic neural field, let us propose the following asynchronous computation model. Each unit of index j is implemented by a task T_j, which calculates the unique solution of the initial value problem of equation (2) at a given simulation time t. At a given sampling time of index i, a subset of tasks $U(i) = \{\cdots T_{j_k} \cdots\}$ whose completion is comprised in the i-th update is defined. This very simple scheme includes synchronous relaxation (i.e., $U(i)$ contains all tasks at each update), deterministic Gauss-Seidel relaxation (i.e., $U(i)$ contains only one task at each update, one after another), other asynchronous schemes (e.g., $U(i)$ contains one or more tasks, randomly drawn with or without replacement), etc.. The original framework is a bit more general, while we decline it in our context only.

Each connection between units of index j and k is supposed to have a sampling delay d_{ijk}, constant or variable, meaning that the information $V_k(t)$ is available to the task j after a delay d_{ijk}. This is different from a simulation delay, meaning that the information $V_j(t)$ is a function of $V_k(t - \delta_{jk}(t))$ though the latter can obviously be simulated in this framework.

As mentioned in the first section, the Mitra computation model is based on two key assumptions: On one hand, all sampling delays $d_{ijk} \leq d < +\infty$ are finite, thus bounded. This first is violated if the link between two tasks is broken. On the other hand, there is a maximal length l between two updates for a given task, i.e. there is no starvation. This second condition is violated with a non-negligible probability in the case of random drawn with replacement, but verified in the other quoted cases.

Based on these reasonable and somehow minimal assumptions, in the linear-case Mitra considers a contracting system, in the sense made explicit in section 3.2, and demonstrates, the *uniform convergence at a geometric rate* of this completely asynchronous numerical scheme. In the non-linear case, Mitra has to assume that the non-linear functions are continuous (but not necessarily smooth) and bounded by a linear contracting function, to obtain the same result. In our case, this means that the non-linear, so-called "sigmoid" function $\sigma()$ can not be a step function, but any other usual continuous profile is convenient.

In other words, as soon as the *sampling* and *simulation* times are not mixed, and the dynamic neural field dynamics attractor is a fixed point, attained with or without damped oscillations, any general asynchronous relaxations schemes with neither unbounded transmission delays, nor starvation, uniformly converge at a geometric rate.

This result has obviously no reason to hold for more complex dynamics, especially in chaotic cases, since even a "negligible" error is going to make the discrete approximation diverge exponentially fast, without any chance to redress the error by a bounded number of asynchronous relaxations. In the case of a periodic stable attractor, though the previous formalism can not be applied as it is, it seems reasonable to assume that the asynchronous relaxations would be able to maintain the discrete approximation scheme at a bounded error of the continuous exact solution.

Thanks to this fundamental result, we can consider the synchronous/asynchronous dynamical neural fields simulation dilemma, (i.e., the fact that authors often wonder whether they efficiently can simulate such a dynamical system using an asynchronous scheme) as solved in such a context.

3.4 An application network example

A unit is defined as before, its state changes only when an event occurs (an update that includes the unit). Let us consider a two-layer network (input and output), each of them being of size $N \times N$ units. The input layer corresponds to the constant current entry that is feeding the output layer. So each unit of index u, v of the output layer receives its input $I_{u,v}$ from the input layer with respect to a receptive field (a Gaussian connection). We suppose that there is neither

lateral connection nor feedback in the input map. Lateral connections in the output map are excitatory, decreasing with distance and the input is inhibitory (the reverse connectivity scheme is also suitable).

It means that each neuron in the input layer is excited by its neighbors and inhibited by the neurons in its receptive field. The resultant activity in the output map is a difference of Gaussian which is a computation scheme very used in dynamic neural networks.

So, in this case, for each unit of the $M = N \times N$ units, equation (2) writes:

$$d/dtV_{u,v}(t) = -V_{u,v}(t) + \sum_{p,q} W_{u,v;p,q} V_{pq}(t) + I_{uv},$$

where $V_{u,v}$ stands for the membrane potential, $W_{u,v;p,q}$ for the weight of the connection from unit (p, q) to unit (u, v) (a positive symmetric tensor in this case), and I_{uv} is the constant current input, projection from the input layer.

If we refer to Mitra's work, this differential equation is a particular case of the proposed framework[2]. Uniform convergence at a geometric rate in the asynchronous mode occurs when, in our case, $\mathbf{I} - |\mathbf{W}|$ is an M-matrix (i.e. a matrix whose off-diagonal entries are less than or equal to zero, with eigenvalues whose real parts are positive). Since $\mathbf{W}$ is a symmetric positive matrix, thus diagonalizable, with positive eigen-values, $\mathbf{I} - |\mathbf{W}|$ is a M-matrix a soon as these eigen-values are smaller than one. As a consequence, we can conclude that, asynchronous computation convergence in such a neural network is guaranteed and that the long-term average rate geometric convergence (per update) is not less than $r = 1/(d + l)$ with r is the spectral radius of $\mathbf{W}$, d is the maximal delay and l is the maximal pseudo-period length.

4 Using discrete event dynamic system

Let us finally consider the other aspect of asynchronous computation, i.e. not the fact that we want to simulate a continuous system in an asynchronous way, but the fact that we want to simulate a system *intrinsically asynchronous*. This covers two fundamental aspects. On one hand, each unit has a local clock, i.e., a local time in some sense, so the state evolution depends entirely on the occurrence of asynchronous mechanisms over time. This means there are delays between units, with unpredictable exact values. On the other hand, there is another semantics related to anachronism, i.e. the fact the information associated to input and output is defined by both some value and the *time* at which such value is issued. In other words, the information is defined through temporal *events*.

As an illustrative case where asynchronous evaluation is not only a matter of simulation but also of modeling, [22] have disclosed pertinent solutions for a discrete beta function (DBF) that correspond to expected biologically plausible responses, though they are not present in the continuous case. This raises the importance of considering asynchronism not only at the implementation level.

Concerning biologically plausible models, the event-driven computation scheme has been mainly developed for spiking neuron models. Such models are not addressed here (see, e.g., [16] for an introduction and [10, 8] for a recent theoretical analysis and general discussion in link with these aspects). Nevertheless, this scheme could be certainly extended at a more mesoscopic level, as that of cortical columns, modeled by dynamical neural fields, as developed here.

The dedicated simulation tool is an event-based neuron simulation kernel as proposed by, e.g., [20] (see [7] for a comparative review) based on the well-known Discrete Event System Specification (DEVS) framework, very easy to simulate on a single processor.

Let us instantiate this general discussion, through an illustrative example. In a recent work [21] a model has been designed that performs global competition, only using local connections, with diffusion of the inhibition throughout the network. This is far quicker to have a few local interactions when computing activity within the network and this makes the model a real candidate for distributed computations. We have re-implemented this mechanism considering asynchronous sampling via a minimal event-based simulation kernel[3], which obviously works since the system is still contracting when using asynchronous sampling, as discussed previously. This has been numerically experimented, as shown in Fig. 2, with the obvious heuristic to have the local sampling period roughly proportional to the state value variation (parsimonious principle), with a string robustness with respect to the related parameters (modifying the asynchronous paradigms changes the transitional values, slightly influences the convergence speed, but does not modify the final result).

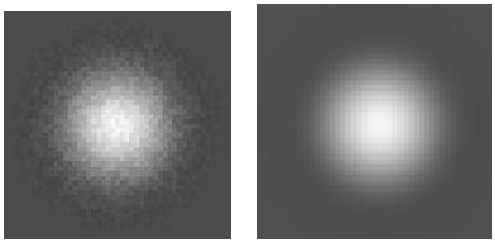

Figure 2. An example of asynchronous sampling of such maps (event-based implementation), applying convergence criteria derived here. We have numerically verified the conjecture that the present results apply when using asynchronous sampling. *Left view*: intermediate result, the fact asynchronous sampling yields randomization is visible. *Right view*: final result, after convergence.

[2] It corresponds to equation (6.7i) [17], with $\mathbf{D} = \mathbf{I}$ the (identity matrix) and $\mathbf{B} = \mathbf{W}$.

[3] Code available at `http://enas.gforge.inria.fr/classNetwork.html`, while `http://mvaspike.gforge.inria.fr` is the general purpose large-scale event-based multi-scale simulator at the edge of the state of the art.

Though this is only a preliminary result it opens large perspectives on new asynchronous paradigms for discrete neural field implementations.

5 Conclusion

By making the distinction between sampling times and simulation times, we have been able to review how well-established asynchronous evaluation methods can be efficiently used for dynamic neural fields simulation; as soon as reasonable assumptions are verified, fast convergence and unbiasedness are guaranteed. In return, as we explained in the previous section, dynamic neural field theory provides a fruitful playground for the study of asynchronous evaluation schemes. For example, in [22], it has been shown (numerically) that such an asynchronous evaluation method leads to novel stable solutions that are functionally very different from the continuous case. When presented with two identical stimuli at different locations, the field is able to stabilize itself on either one of the two stimuli, hence breaking the symmetry of the system. However, this new state, that has been shown to be very stable, can be also easily proved not to be a solution of the continuous equation of the field. What is thus the relevancy of such a continuous description if we are to evaluate it using numerical asynchronous equations ? Ideally, we wish we could have an equivalent continuous asynchronous description but unfortunately, this is not yet the case in the field of mathematics. We should then take extra precaution when describing a system using continuous equations and wonder if we are really simulating what we advertised in the definition of the system. Particularly, at the mesoscopic modeling level, it may be worthwhile to use an event-based paradigm instead of a clock-based one, as it is a well-defined paradigm which takes into consideration that not only the processing but also the timing are fully distributed.

From a more cognitive point of view, this study reveals the implicit presence of a central clock in a number of models and thus the implicit presence of a grand supervisor (a.k.a. central executive, homunculus, etc.) orchestrating the overall activity of the model. While this may be acceptable in most models that do not care about this parasitic presence, it is hardly acceptable if a model pretends to vanquish the curse of the homunculus.

Acknowledgment: This work was partially supported by the ANR Project MAPS and benefited from helpful discussions with Axel Hutt and Hervé Frezza-Buet.

References

[1] F. Alexandre, J. Fix, N. Rougier, and T. Viéville. Algorithmic adjustment of neural field parameters. Research Report RR-6923, INRIA, 2009.

[2] S. Amari. Dynamic of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics*, 27:77–88, 1977.

[3] J. Bahi and S. Contassot-Vivier. A convergence result on fully-asynchronous discrete-time discrete-state dynamic networks. Research report, AND Team, LIFC, IUT de Belfort-Montbliard, Belfort, France, 2002.

[4] C.L. Barret and C.M. Reidys. Elements of a theory of computer simulation i: Sequential ca over random graphs. *Applied Mathematics and Computation*, 98:241, 1999.

[5] D.P. Bertsekas and J.N. Tsitsiklis. Some aspects of parallel and distributed iterative algorithms - a survey. *Automatica*, 27:3–21, 1991.

[6] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[7] R. Brette, M. Rudolph, T. Carnevale, M. Hines, D. Beeman, J.M. Bower, M. Diesmann, A. Morrison, P. H. Goodman, F. C. Harris Jr., M. Zirpe, T. Natschläger, D. Pecevski, B. Ermentrout, M. Djurfeldt, A. Lansner, O. Rochel, T. Vieville, E. Muller, A.P. Davison, S. El Boustani, and A. Destexhe. Simulation of networks of spiking neurons: a review of tools and strategies. *Journal of Computational Neuroscience*, 23(3):349–398, 2007.

[8] B. Cessac, H. Paugam-Moisy, and T. Viéville. Overview of facts and issues about neural coding by spikes. *J. Physiol. Paris*, 104(1-2):5–18, 2010.

[9] B. Cessac and M. Samuelides. From neuron to neural networks dynamics. *EPJ Special topics: Topics in Dynamical Neural Networks*, 142(1):7–88, 2007.

[10] B. Cessac and T. Viéville. On dynamics of integrate-and-fire neural networks with adaptive conductances. *Frontiers in neuroscience*, 2(2), jul 2008.

[11] D. Chazan, W. Miranker, and T. Viéville. Chaotic relaxation. *Linear Algebra and its Applications*, 2:199–222, 1969.

[12] S. Chemla, F. Chavane, T. Vieville, and P. Kornprobst. Biophysical cortical column model for optical signal analysis. In *Sixteenth Annual Computational Neuroscience Meeting (CNS)*, jul 2007.

[13] N.A. Fates. Asynchronism induces second order phase transitions in elementary cellular automata. *Journal of Cellular Automata*, 4:21–28, 2009.

[14] N.A. Fates and M. Morvan. An experimental study of robustness to asynchronism for elementary cellular automata. *Complex Syst.*, 16:1–27, 2005.

[15] L.D. Garcia, A.S. Jarrah, and R. Laubenbacher. Sequential dynamical systems over words. *Applied Mathematics and Computation*, 174:500–510, 2006.

[16] W. Gerstner and W.M. Kistler. *Spiking Neuron Models*. Cambridge University Press, 2002.

[17] D. Mitra. Asynchronous relaxations for the numerical solution of differential equations by parallel processors. *SIAM J. Sci. Stat. Comput.*, 8(1):43–58, 1987.

[18] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, 1988.

[19] F. Robert. *Les systèmes dynamiques discrets*, volume 19. Springer, 1994.

[20] O. Rochel and D. Martinez. An event-driven framework for the simulation of networks of spiking neurons. In *Proc. 11th European Symposium on Artificial Neural Networks*, pages 295–300, 2003.

[21] N. Rougier. Dynamic neural field with local inhibition. *Biological Cybernetics*, 94(3):169–179, 2006.

[22] N. Rougier and J. Vitay. Emergence of attention within a neural population. *Neural Networks*, 19(5):573–581, 2006.

[23] J.G. Taylor. Neural bubble dynamics in two dimensions: foundations. *Biological Cybernetics*, 80:5167–5174, 1999.

[24] S. Vahie and N. Jouppi. Dynamic neuronal ensembles: A new paradigm for learning and control. *AI, Simulation and Planning in High Autonomy Systems*, 1996.

[25] T. Viéville, S. Chemla, and P. Kornprobst. How do high-level specifications of the brain relate to variational approaches? *J. Physiol. Paris*, 101, 2007.

[26] L. Watts. Event-driven simulation of networks of spiking neurons. *Advances in neural information processing systems*, 6:927–934, 1994.

[27] H.R. Wilson and J.D. Cowan. A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Kybernetic*, 13:55–80, 1973.

[28] B. Zeigler, T. Kim, and H. Praehofer. *Theory of modeling and simulation*. Academic Press, 2 edition, 2000.